



[Home](#) » [Articles](#) » [Gaming](#) » Beginner

How To Get Started with Gaming AI

By [James Matthews](#)

Gaming AI is definitely one of the most interesting areas of Artificial Intelligence, since it can have immediately noticeable effects. Yet, the biggest problem with programming game AI, is that you often have to program a game to go with it! This is too much trouble for many people, so this article will help you look at various ways to overcome these difficulties.

This article will look at several approaches: a non-programming approach, a programming approach using existing games, and programming from scratch.

No Programming: MindRover

MindRover is a game produced by CogniToy (see [Generation5 review](#)) that allows a player to choose a rover then add components and wire it up to perform a certain task: whether it is to blow something up, win a race, navigate a circuit or all three!

MindRover is an excellent way to experiment with game AI, because it assumes *no programming* knowledge. All the programming is done via a graphical user interface (shown on the right). For example, once you have picked your vehicle, you can add radars, steering components and weapons. Then, you can wire the components up so that when the radar detects something, they will fire the weapon and steering the direction of the detected object.



So you run your scenario and find that your rover is alright, but if the opponent fires a slow moving rocket, the radar will pick it up and turn towards it. This is not really what you want, so you go back and add an IFF (identification friend or foe) filter on your radar, and filter out anything that is not an enemy. Run the scenario and your rover now works much better.

MindRover robots can be much more complicated than this, allowing for many components (perhaps as many as 30). These can be interconnected using logic gates, broadcasts (one event triggers many), mode switches (events will only happen when a certain mode is set - evade and pursuit behaviour perhaps), and many other things.

MindRover augments this by simply being a graphical front-end to a complete programming language called ICE. Below is a clipping of the code that was generated from the rover shown in the above screenshot.

```
class DkrPursuit extends Vehicle
  sys as VehicleSystemObject

  Filter_IFF1 as Filter_IFF
  MediumEngine1 as MediumEngine
  Steering1 as Steering
  BumpSensor1 as BumpSensor
  Filter_IFF2 as Filter_IFF
  LongRangeRadar1 as LongRangeRadar
  MediumRadar1 as MediumRadar
  MediumRadar2 as MediumRadar
  MediumRadar3 as MediumRadar
  MediumRadar4 as MediumRadar
  Broadcast1 as Broadcast
  TurnLeft as Broadcast
  Timer1 as Timer

  ' class functions are here...

end class 'DkrPursuit
```

While ICE is completely beyond the scope of this document, it is definitely something worth noting and experimenting

with, since you can do things in ICE you obviously cannot do with the interface. In fact, there is a growing community of ICE programmers that have added additional components to the game, such as more advanced logic gates, components that emulate EEPROM memory and many more things. Therefore, if you do not have a strong programming language, and what something cool to happen quickly, take a look at MindRover. As you get more interested, you can read up on ICE and program additional components and more intelligent rovers.

Creating AI Mods

With the advent of games such as Quake II, game developers saw the use of making games easily expandable. Quake II had all of its game code in a DLL, which it loaded at runtime. iD Software (makers of the Quake series) released the source code for the game code, allowing anybody to modify the weapons, gameplay aspects, physics and most importantly (for us) the artificial intelligence. It is relatively easy to start modifying AI for games such as this, because it can begin with tweaking numbers and progressing to modifying AI functionality and finally to implementing new AI features.

You will generally need C or C++ programming skills and a decent compiler that supports Win32 DLLs. Microsoft Visual C++ is easily the best compiler for this duty, although its price tag puts it out of most hobby budgets. There are other alternatives (see the external links section for more information) that will suffice, producing just as nice code but without the funky graphical interface.

The screenshot here shows one of the results of my Quake II mod that improved on the artificial intelligence quite significantly. Here you see a soldier jumping over a shot aimed at his legs, and firing while he is in the air. The RealAI mod had these features:



- Tweaked ducking function. The enemy ducked only as long as was necessary.
- Some enemies jumped when shots were fired at their feet.
- Monsters could swim. Default Q2 monsters stay out of the water, but the RealAI monsters could swim and would come up for air when necessary.
- The monsters will not shoot when one of their own kind is in front of them. This includes strafing monsters.
- Experimented with shooting exploboxes.
- Experimented with pathing algorithms.

The joy of modifying the Q2 source code is that you already have an excellent game, with excellent graphics, many maps and gameplay. Therefore, by tweaking the AI to your wants, you can have your ultimate game. Other games such as Unreal, HalfLife and Quake III also allow you to customize the game to this depth. So, if you have any of these games, know C/C++ and have a decent compiler - get coding!

From Scratch...

Coding game AI from scratch often means a lot of overhead with the other areas of the game. Therefore, board games are an excellent choice for coding game AI, since they have very simple graphic requirements (which can also often be reused in other board games) and the rules are a necessary part of the AI. Here are some ideas:

- Pente (medium)
- Tic-Tac-Toe (easy)
- Othello (easy-medium)
- Backgammon (medium)
- Chess (hard)
- Connect 4 (easy)
- Go (hard)
- 3D Tic-Tac-Toe (easy-medium)

Programming board games is great as a starting AI project, since impressive opponents can be created by using simple heuristics (see [Simple Board Game AI](#)) without any complicated AI techniques. You can perhaps extend your board game AI with game trees and a [minimax](#) algorithm. The benefit with creating your own game from scratch is it can be as complicated or as simple as you wish, and you can constantly modify it. Be warned though that even board game programming can be tedious to complete - do not set your sights too high. Complete your project a step at a

time.

Conclusion

Gaming AI requires you to either be able to program an entire game, or modify a commercial game to your tastes. Games such as MindRover are designed for AI modification from the beginning, which makes it an excellent start. If you want more impressive results (in terms of graphics etc.) then modifying one of the newer FPS-type games is often much harder but has some great benefits. If you do want to code the entire game yourself, something simple like a board game is a great place to start.

Last Updated: 03/02/2001

Article content copyright © James Matthews, 2001.

All content copyright © 1998-2007, Generation5 unless otherwise noted.

- [Privacy Policy](#) - Legal - Terms of Use -